Answering Similarity Queries in Peer-to-Peer Networks

Panos Kalnis, Wee Siong Ng, Beng Chin Ooi and Kian-Lee Tan
Department of Computer Science
National University of Singapore
www.comp.nus.edu.sg/~{kalnis, ngws, ooibc, tankl}

Categories and Subject Descriptors: H.2.4:Systems:Distributed databases
General Terms: Design
Keywords: Peer-to-Peer, Image, Similarity

1. INTRODUCTION

Broadcast-based systems Peer-to-Peer (P2P) systems use message-flooding to propagate queries and have been successfully employed in practice to form large-scale ad-hoc networks. Most existing systems (e.g. Gnutella) support only keyword searching. Each file is characterized by its metadata and queries ask for combinations of keywords. Consider for instance a music sharing system. Users ask for a song title, or a combination of an artist and album name. Such queries can be unambiguously evaluated as "found" or "not found" by searching the metadata for matching keywords.

In this paper we investigate a different problem: Users ask fuzzy queries like "find the top-k images which are similar to a given sample". Such queries are common in image retrieval systems. Since there is no centralized index, each peer within the query horizon is contacted and returns k results (i.e., the top-k local images) to the initiator, which, in turn, computes the global result. Unfortunately, the extremely low selectivity of such queries floods the network with useless messages. Alternatively we could set a threshold similarity and accept answers only above this value. The issue in this case is how to select the query-dependant threshold given that the interpretation of an image depends on the user's perception. Moreover, this method would not reduce the number of query messages which grows exponentially with the number of hops.

Observe, however, that due to the fuzzy nature of the queries, the answers are always approximations. As a result, if two queries are similar, the top-k answers for the first one may contain (with high probability) some of the answers for the second query. In addition, in P2P networks each peer can examine the messages that pass through it. Motivated by these observations, we developed *FuzzyPeer*, a generic P2P system which supports similarity queries. In FuzzyPeer some of the queries are *frozen* (i.e., they are not propagated further) inside a set of peers. The frozen queries are answered by the stream of results that passes through the peers, and was initiated by the remaining running queries. By carefully selecting the set of frozen queries, the quality of the results and the response time remain at acceptable levels even when the system is overloaded. Additionally, the number of messages drops considerably, thus improving the scalability and the throughput of the network. Moreover, our optimization algorithms do not pose any overhead due to synchronization messages.

WWW2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

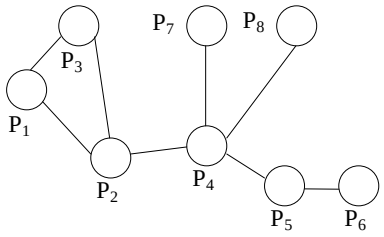


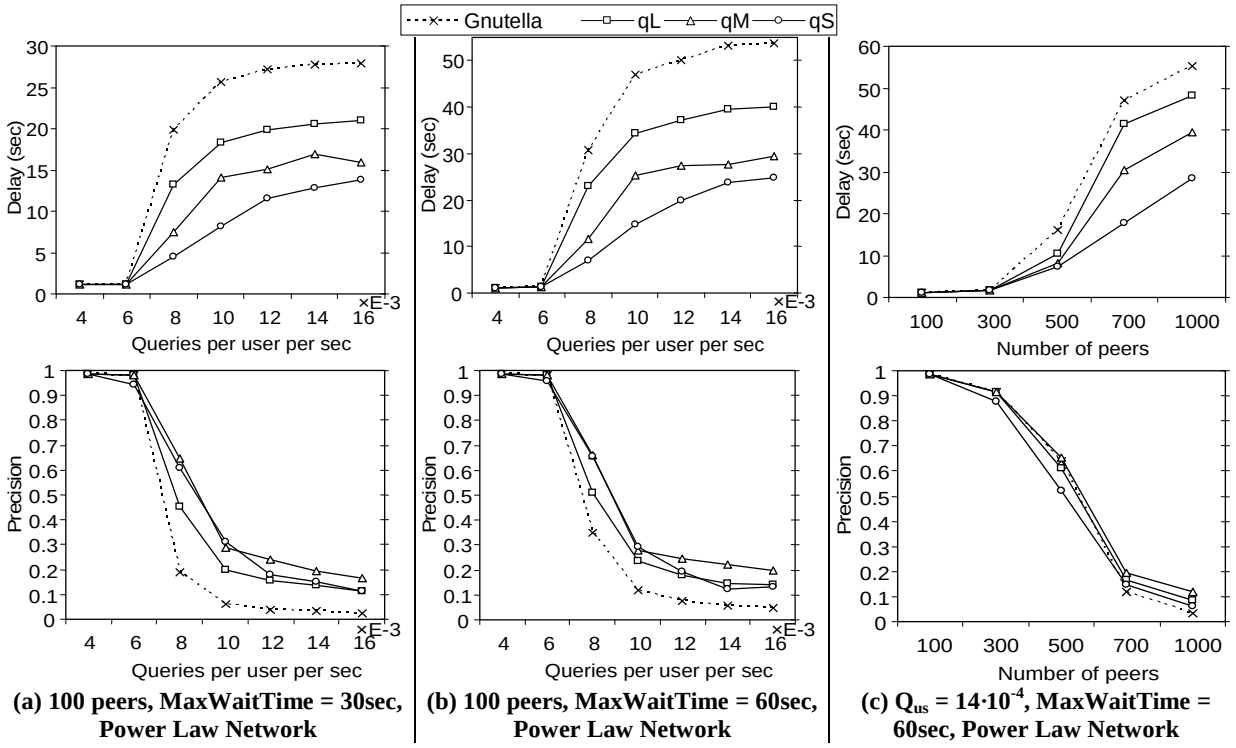
Figure 1: A Typical FuzzyPeer network

Although throughout this paper we focus on image retrieval, our methods are applicable to other domains where similarity searching is performed in P2P networks; as an example, consider the case of text retrieval. Moreover, the network topology does not need to be flat. For instance, given a two-level super-peer organization (e.g., Morpheus) we can apply the same techniques at the upper level which contains the index of its clients, rendering the entire system more scalable.

2. SYSTEM DESCRIPTION

Figure 1 depicts a typical FuzzyPeer network. Let the user of P_1 ask a query q: "find the top-10 images which are similar to a given sketch". P_1 will broadcast q to P_2 and P_3. The receiving nodes will search their databases and return the ids of the top-10 most similar images together with a similarity measure to P_1. At the same time they will broadcast q to their neighbors. For example, P_2 will send q to P_3 (which will reject the duplicate message) and P_4. Notice that P_4 will return the results through P_2. Queries can propagate for up to a maximum number of hops d. Assuming that $d = 3$, the query cannot reach P_6. P_1 waits for up to $MaxWaitTime$; during this interval it receives the answers and continuously refines the result. After $MaxWaitTime$ expires, any answer message that reaches P_1 is rejected.

Assume now that soon after P_1, P_3 also submits a query q' which is similar to q. In a traditional P2P system q' propagates through P_2 the same way as q. q' causes messages to pass through P_2 almost simultaneously with the messages generated by q. Therefore, P_2 is overloaded and all messages are delayed. If the delay is long enough, $MaxWaitTime$ expires causing q and q' to terminate before they receive enough useful results. Notice, however, that we can do better: When q passes through P_2 it initiates an answer stream $Stream_q$. All the answers from P_4, P_5, P_7 and P_8 will go through $Stream_q$. When q' reaches P_2 the system can identify that q and q' are similar, so instead of been propagated, q' will freeze inside P_2 and will be attached to $Stream_q$. P_2 will afterwards duplicate and sent to P_3 all answers that reach $Stream_q$.



(a) 100 peers, MaxWaitTime = 30sec, Power Law Network

(b) 100 peers, MaxWaitTime = 60sec, Power Law Network

(c) Q_{us} = 14·10^{-4}, MaxWaitTime = 60sec, Power Law Network

Figure 2: Adaptive Freezing Algorithm. First row: $Delay$ until receiving the first result. Second row: $Precision$.

There are several benefits of this approach: (i) Thrashing is avoided. Instead of not answering any query at all, a considerable percentage of the queries can locate accurate answers. (ii) Excess queries are frozen instead of aborted. Since all answers are approximations, there is a high probability for a frozen query to receive accurate results if it attaches to a similar stream. This is different from other systems (e.g. web search engines) where the probability of finding a concurrent similar query is low. In such systems queries ask for certain keywords and run in the server for a few msec, while in P2P systems queries run for around 3 orders of magnitude more time (i.e., 100's of sec) (iii) Even if the results for the frozen queries are not accurate, users can utilize them to refine their original query.

Our Adaptive Query Freezing (AQF) algorithm is described in details in the full version of this paper [1].

3. EXPERIMENTAL EVALUATION

We employed two implementations to evaluate our methods. The first one is a JAVA prototype based on our BestPeer platform [2]; it was used to derive the basic parameters of the system). The parameters were used in our simulator. Our dataset consisted of a library of 10504 high resolution images. We used a vector of 48 coefficients to represent the visual features of each image.

Here, we evaluate our freezing algorithm *AQF*. Figure 2 presents the results for power-law networks with varying query rate Q_{us}. AQF uses a parameter a_q to control the length of message queues in the peers and freeze queries accordingly. We use three settings which result to long *qL*, medium *qM* and short queues *qS*. The performance is compared against a Gnutella-style broadcast based approach, without query freezing.

We use two metrics to evaluate the algorithms: (i) the $Delay$ until the arrival of the first *useful* result and (ii) the $Precision$ of the final result set. When the query rate is low, all methods are equivalent. However, as the query rate increases, AQF clearly outperforms the Gnutella-style method both in terms of delay and precision. The behavior is similar when we increase the number of peers in the network (Figure 2.c). Clearly, our method improves the scalability of the system in terms of nodes and query throughput.

4. CONCLUSION

In this paper we dealt with the problem of retrieving information from large repositories built on top of ad-hoc P2P networks. While most existing approaches are limited to exact key matching, we developed FuzzyPeer which supports content based similarity queries. Due to the absence of centralized indexing, such queries are challenging; the difficultly of defining an application independent terminating criterion in addition to their extremely low selectivity, overload the system with useless messages and cause thrashing. We addressed this issue by introducing the freezing technique: some queries are paused and attached to answer streams from similar concurrently running ones, since the answer for both queries is expected to overlap. We proposed AQF, a simple yet efficient distributed optimization algorithm which improves the scalability and the throughput of the system. Numerous applications, including full-text searching in large archives or fuzzy queries in distributed multimedia repositories, can benefit from our techniques as we demonstrated by an image retrieval case study.

5. REFERENCES

[1] P. Kalnis, W. S. Ng, B. C. Ooi, and K. L. Tan. Answering similarity queries in peer-to-peer networks. http://www.comp.nus.edu.sg/-kalnis/ASQ.pdf.

[2] W. Ng, B. Ooi, and K. Tan. Bestpeer: A self-configurable peer-to-peer system (poster). In *ICDE*, 2002.